Seasonality, Holiday Effects, And Regressors

**Modeling Holidays and Special Events**

If you have holidays or other recurring events that you’d like to model, you must create a dataframe for them. 对于你需要模拟的节假日和特殊事件，你都需要为他们预先创建一个新的dataframe

**It has two columns** (holiday and ds) and a row for each occurrence of the holiday. 它有两列(holiday和ds)和一表示每个出现的节日的行。它必须包括假期发生的所有事件，包括过去(从历史数据开始)和未来(从预测开始)。

It must include all occurrences of the holiday, both in the past (back as far as the historical data go) and in the future (out as far as the forecast is being made). 它必须包括节日发生的所有事件，包括过去的(追溯到历史数据)和未来的(直到做出预测)。

If they won’t repeat in the future, Prophet will model them and then not include them in the forecast. 如果它们在未来不会重复，Prophet模型将自动对它们进行建模，然后不将它们包括在预测中。

You can also include columns lower\_window and upper\_window which extend the holiday out to [lower\_window, upper\_window] days around the date. 你还可以设置包括列lower\_window和upper\_window，它们将假期扩展到[lower\_window, upper\_window]附近的日期。**即你可设置此节假日事件的影响范围**

For instance, if you wanted to include Christmas Eve in addition to Christmas you’d include lower\_window=-1,upper\_window=0. 例如，如果想在圣诞节之外包含平安夜，需要包含lower\_window=-1,upper\_window=0。（圣诞节前一天是平安夜）

If you wanted to use Black Friday in addition to Thanksgiving, you’d include lower\_window=0,upper\_window=1. You can also include a column prior\_scale to set the prior scale separately for each holiday, as described below. 如果想在感恩节之外使用黑色星期五，就需要包含**lower\_window=0,upper\_window=1**。（感恩节的第二天是黑色星期五）你也可以包含一个列**prior\_scale**来分别设置每个假期的先验比例，如下所示。

Here we create a dataframe that includes the dates of all of Peyton Manning’s playoff appearances:

1. # Python
2. playoffs = pd.DataFrame({
3. 'holiday': 'playoff',

4 'ds': pd.to\_datetime(['2008-01-13', '2009-01-03', '2010-01-16',

5 '2010-01-24', '2010-02-07', '2011-01-08',

6 '2013-01-12', '2014-01-12', '2014-01-19',

|  |  |  |
| --- | --- | --- |
| 7 |  | '2014-02-02', '2015-01-11', '2016-01-17', |
| 8 |  | '2016-01-24', '2016-02-07']), |
| 9 | 'lower\_window': 0, |  |
| 10 | 'upper\_window': 1, |  |
| 11 }) |  |  |

12 superbowls = pd.DataFrame({

13

14

15

16

17 })

'holiday': 'superbowl',

'ds': pd.to\_datetime(['2010-02-07', '2014-02-02', '2016-02-07']),

'lower\_window': 0,

'upper\_window': 1,

18 holidays = pd.concat((playoffs, superbowls))

Above we have included the superbowl days as both playoff games and superbowl games. This means that the superbowl effect will be an additional additive bonus on top of the playoff effect. 这意味着超级碗效应将是季后赛效应之外的额外影响。

Once the table is created, holiday effects are included in the forecast by passing them in with the holidays argument. Here we do it with the Peyton Manning data from the [Quickstart](https://facebook.github.io/prophet/docs/quick_start.html):

1. # Python
2. m = Prophet(holidays=holidays)
3. forecast = m.fit(df).predict(future)

The holiday effect can be seen in the forecast dataframe:节假日的印象可以在 **forecast** dataframe中查询

1. # Python
2. forecast[(forecast['playoff'] + forecast['superbowl']).abs() > 0][
3. ['ds', 'playoff', 'superbowl']][-10:]

**2190**2014-02-02 1.223965 1.201517

**2191**2014-02-03 1.901742 1.460471

**2532**2015-01-11 1.223965 0.000000

**2533**2015-01-12 1.901742 0.000000

**2901**2016-01-17 1.223965 0.000000

**2902**2016-01-18 1.901742 0.000000

**2908**2016-01-24 1.223965 0.000000

**2909**2016-01-25 1.901742 0.000000

**2922**2016-02-07 1.223965 1.201517

**2923**2016-02-08 1.901742 1.460471

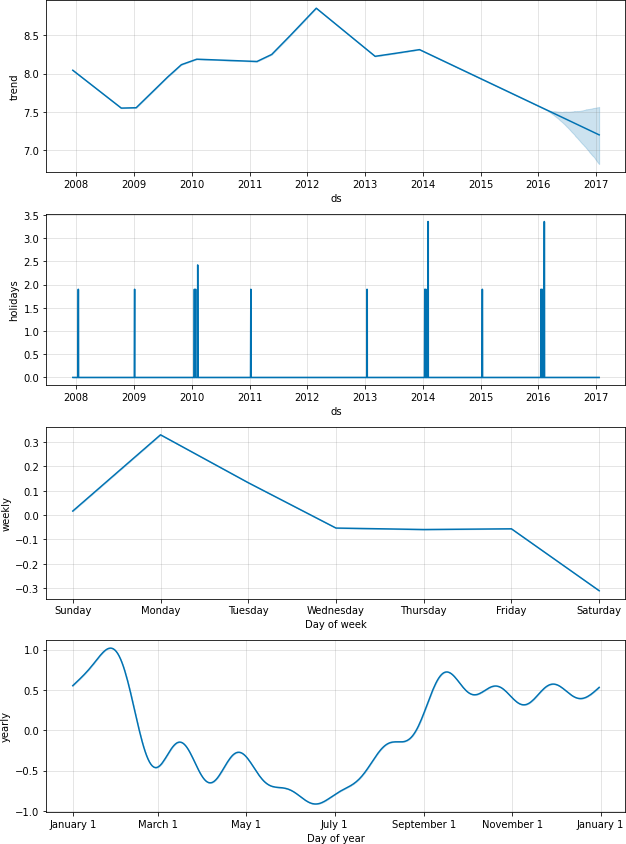
**superbowl**

**playoff**

**ds**

The holiday effects will also show up in the components plot, where we see that there is a spike on the days around playoff appearances, with an especially large spike for the superbowl:

1. # Python
2. fig = m.plot\_components(forecast)
3. plt.show()



Individual holidays can be plotted using the **plot\_forecast\_component** function (imported from **prophet.plot** in Python) like **plot\_forecast\_component(m, forecast, 'superbowl')** to plot just the superbowl holiday component. 可以使用**plot\_forecast\_component函数**(从prophet导入)绘制单个假日。

**调用此函数返回值为一个matplotlib.artist（列表类型）后边增加plt.show()函数才显示**

# Built-in Country Holidays(内置的国家假期)

You can use a built-in collection of country-specific holidays using the ***add\_country\_holidays*** method (Python) or function (R). 你可以使用***add\_country\_holidays***方法

The name of the country is specified, and then major holidays for that country will be included in addition to any holidays that are specified via the ***holidays*** argument described above:

除了之前示例中我们可以自定义一些***holidays***参数以外，还可以指定某国家的主要假期来使用Prophet中的设置。

1. # Python
2. m = Prophet(holidays=holidays)
3. m.add\_country\_holidays(country\_name='US')
4. m.fit(df)

You can see which holidays were included by looking at the ***train\_holiday\_names*** (Python) attribute of the model:

你可以通过模型的***train\_holiday\_names*** (Python)

1. # Python
2. m.train\_holiday\_names
3. 0 playoff
4. 1 superbowl
5. 2 New Year's Day
6. 3 Martin Luther King Jr. Day
7. 4 Washington's Birthday
8. 5 Memorial Day
9. 6 Independence Day
10. 7 Labor Day
11. 8 Columbus Day
12. 9 Veterans Day
13. 10 Thanksgiving
14. 11 Christmas Day
15. 12 Christmas Day (Observed)
16. 13 Veterans Day (Observed)
17. 14 Independence Day (Observed)
18. 15 New Year's Day (Observed)
19. dtype: object

The holidays for each country are provided by the holidays package in Python.每个国家的假日由Python中的假日包提供。

A list of available countries, and the country name to use, is available on their page 以下是有关于此包的详细信息: [***https://github.com/dr-prodigy/python-holidays***](https://github.com/dr-prodigy/python-holidays).

In addition to those countries, Prophet includes holidays for these countries: Brazil (BR), Indonesia (ID), India (IN), Malaysia (MY), Vietnam (VN), Thailand (TH), Philippines (PH), Pakistan (PK), Bangladesh (BD), Egypt (EG), China (CN), and Russian (RU), Korea (KR), Belarus (BY), and United Arab Emirates (AE).

In Python, most holidays are computed deterministically and so are available for any date range; 在Python中，大多数假期的计算都是确定的，因此适用于任何日期范围

a warning will be raised if dates fall outside the range supported by that country. In R, holiday dates are computed for 1995 through 2044如果日期超出该国支持的范围，将发出警告

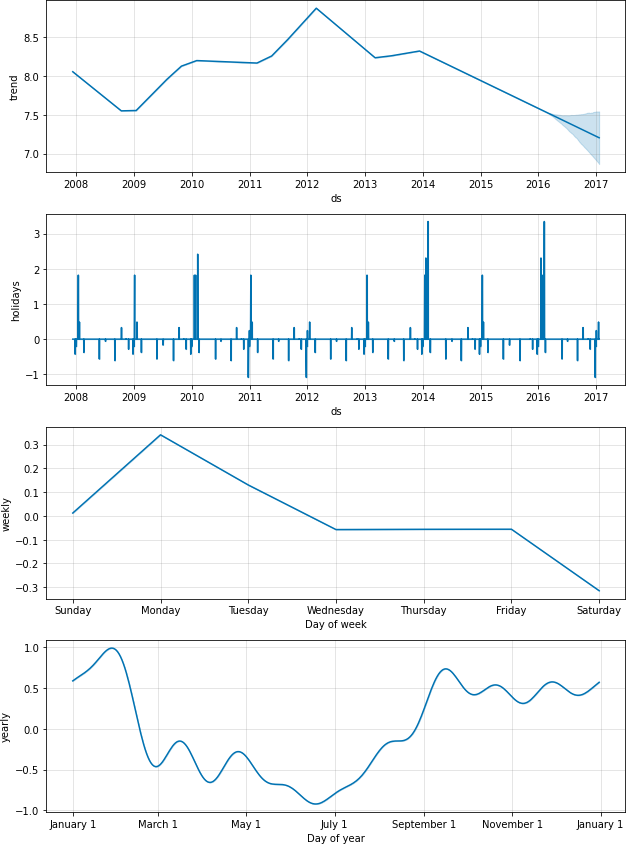
。

and stored in the package as ***data-raw/generated\_holidays.csv***. If a wider date range is needed, this script can be used to replace that file with a different date range: ***https://github.com/facebook/prophet/blob/main/python/scripts***

/generate\_holidays\_file.py. 如果需要更大的日期范围，则可以使用此脚本将该文件替换为不同的日期范围:**https://github.com/facebook/prophet/blob/main/python/scripts/generate\_holidays\_file.py。**

As above, the country-level holidays will then show up in the components plot:将countryholidays画图显示

1. # Python
2. forecast = m.predict(future)
3. fig = m.plot\_components(forecast)



# Fourier Order for Seasonalities

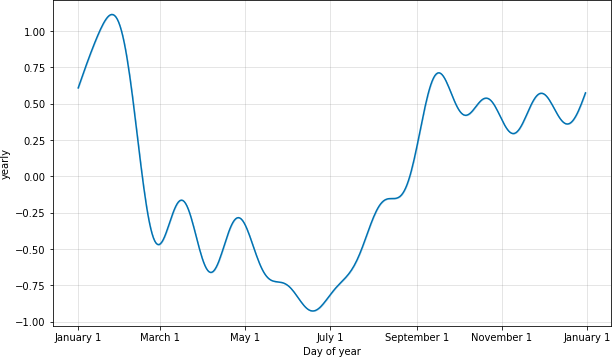
Seasonalities are estimated using a partial Fourier sum. See [the paper](https://peerj.com/preprints/3190/) for complete details, and this figure on Wikipedia for an illustration of how a partial Fourier sum can approximate an arbitrary periodic signal. 估计季节性使用了部分的傅里叶加法，详细信息请参阅论文，维基百科上的这张图说明了傅里叶部分和如何近似任意周期信号。

**The number of terms in the partial sum (the order) is a parameter that determines how quickly the seasonality can change. 部分和中的项数(顺序)是一个参数，它决定了季节性变化的快慢。**

To illustrate this, consider the Peyton Manning data from the [Quickstart](01%20安装、快速启动.docx). The default Fourier order for yearly seasonality is 10, which produces this fit:在[Quick\_Start](01%20安装、快速启动.docx)中Peyton Manning实验中，年季节性的默认傅里叶顺序是10,他产生的拟合效果如下图。

1. # Python
2. from prophet.plot import plot\_yearly 3 m = Prophet().fit(df)

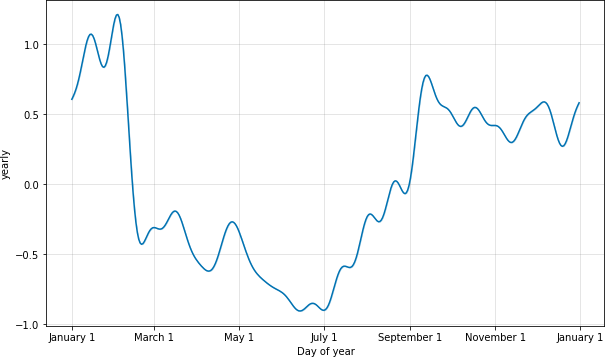
4 a = plot\_yearly(m)



**The default values are often appropriate, but they can be increased when the seasonality needs to fit higher-frequency changes, and generally be less smooth. 默认值通常是合适的，但当季节性需要适应更高频率的变化时，可以增加默认值，而且通常不太平滑。**

The Fourier order can be specified for each built-in seasonality when instantiating the model, here it is increased to 20: 在实例化模型时，可以为每个内置季节性指定傅里叶级数，这里增加到20:

1. # Python
2. from prophet.plot import plot\_yearly
3. m = Prophet(yearly\_seasonality=20).fit(df) 4 a = plot\_yearly(m)



Increasing the number of Fourier terms allows the seasonality to fit faster changing cycles, but can also lead to overfitting: N Fourier terms corresponds to 2N variables used for modeling the cycle

增加傅里叶项的数量可以使季节性更适合变化更快的周期，**但也可能导致*过拟合***:

**N个傅里叶项对应于用于建模周期的2N个变量**

# Specifying Custom Seasonalities

Prophet will by default fit weekly and yearly seasonalities, if the time series is more than two cycles long. It will also fit daily seasonality for a sub-daily time series. You can add other seasonalities (monthly, quarterly, hourly) using the ***add\_seasonality*** method (Python) or function (R).默认情况下，当数据中的时间序列超过两个周期，Prophet将会在每个周期内自动训练（通过Prophet.fit()方法）周季节性、年季节性。当数据只以日为单位时，它还将训练（通过Prophet.fit()方法）每日季节性也可以通过函数 ***add\_seasonality()***方法自定义添加你想要的季节性（例如：每月、每季度、每小时等）

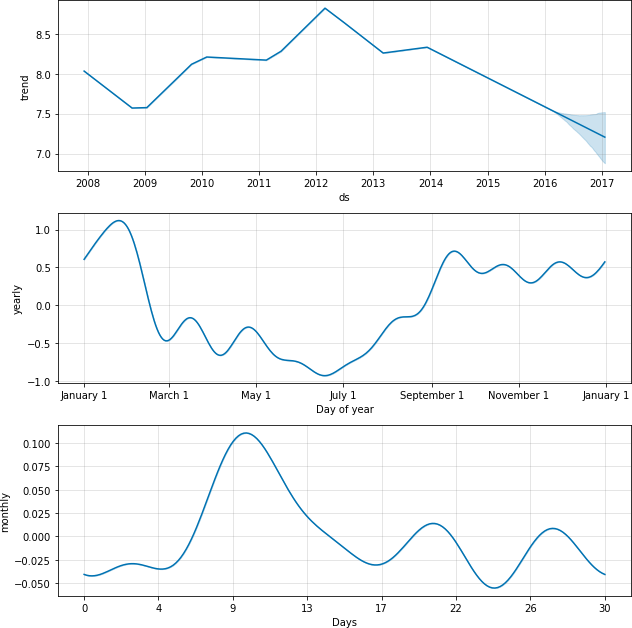
The inputs to this function(***add\_seasonality()***) are a name, the period of the seasonality in days, and the Fourier order for the seasonality. ***add\_seasonality()***的输入包含三个方面，自定义的季节性名称、以天为单位的季节周期(小时周期换算为x/24天)

For reference, by default Prophet uses a Fourier order of 3 for weekly seasonality and 10 for yearly seasonality. An optional input to ***add\_seasonality*** is the prior scale for that seasonal component - this is discussed below. 作为参考，默认情况下Prophet使用3的傅里叶order表示周季节性，10的傅里叶order表示年季节性。该季节组成部分的先验尺度是***add\_seasonality***的一个可选输入——这将在下面讨论。

As an example, here we fit the Peyton Manning data from the [Quickstart](https://facebook.github.io/prophet/docs/quick_start.html), but replace the weekly seasonality with monthly seasonality. The monthly seasonality then will appear in the components plot: 例如，这里我们拟合了来自Quickstart的Peyton Manning数据，但将每周的季节性替换为每月的季节性。然后每月的季节性将出现在组成部分的图中:

1. # Python
2. m = Prophet(weekly\_seasonality=False)
3. m.add\_seasonality(name='monthly', period=30.5, fourier\_order=5) 4 forecast = m.fit(df).predict(future)

5 fig = m.plot\_components(forecast)



# Seasonalities that depend on other factors

# 取决于其他因素的季节性

In some instances the seasonality may depend on other factors, such as a weekly seasonal pattern that is different during the summer than it is during the rest of the year, or a daily seasonal pattern that is different on weekends vs. on weekdays. 在某些情况下，季节性可能取决于其他因素，例如在夏季与一年其余时间不同的每周季节性模式，或者在周末与工作日不同的每日季节性模式。

These types of seasonalities can be modeled using conditional seasonalities.

这些类型的季节性可以用条件季节性来建模。

Consider the Peyton Manning example from the [Quickstart](https://facebook.github.io/prophet/docs/quick_start.html). The default weekly seasonality assumes that the pattern of weekly seasonality is the same throughout the year, but ***we’d expect the pattern of weekly seasonality to be different during the on-season (when there are games every Sunday) and the off-season***. 默认的周季节性假设全年的周季节性模式是相同的，**但我们预计周季节性模式在赛季期间(当每个周日都有比赛时)和非赛季期间是不同的。**

We can use **conditional seasonalities** to construct separate on-season and off-season weekly seasonalities. 我们可以使用条件季节性来构建单独的季节和淡季周季节性

First we add a boolean column to the dataframe that indicates whether each date is during the on-season or the off- season:

首先，我们向dataframe中添加一个布尔值列，表示每个日期是淡季还是旺季:

1. # Python
2. def is\_nfl\_season(ds):
3. date = pd.to\_datetime(ds)
4. return (date.month > 8 or date.month < 2) 5
5. df['on\_season'] = df['ds'].apply(is\_nfl\_season)
6. df['off\_season'] = ~df['ds'].apply(is\_nfl\_season)

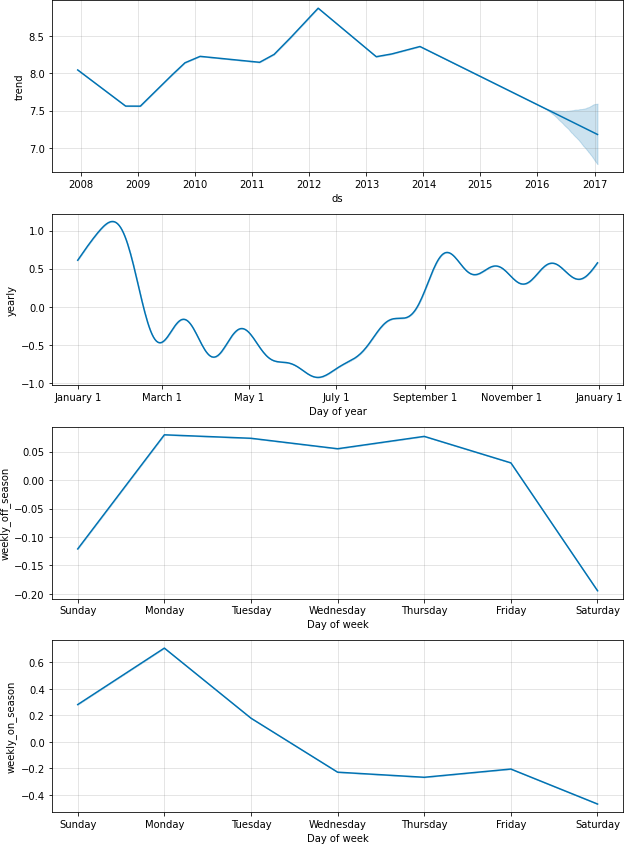
Then we disable the built-in weekly seasonality, and replace it with two weekly seasonalities that have these columns specified as a condition. 然后，我们禁用内置的每周季节性，并将其替换为两个指定这些列作为条件的每周季节性。

This means that the seasonality will only be applied to dates where the ***condition\_name*** column is ***True***. 这意味着季节性只会应用于condition\_name列为True的日期。

We must also add the column to the ***future*** dataframe for which we are making predictions. 我们还必须将该列添加到我们正在进行预测的未来数据框中。

1. # Python
2. m = Prophet(weekly\_seasonality=False)
3. m.add\_seasonality(name='weekly\_on\_season', period=7, fourier\_order=3, condition\_name='on\_season')
4. m.add\_seasonality(name='weekly\_off\_season', period=7, fourier\_order=3, condition\_name='off\_season') 5
5. future['on\_season'] = future['ds'].apply(is\_nfl\_season)
6. future['off\_season'] = ~future['ds'].apply(is\_nfl\_season) 8 forecast = m.fit(df).predict(future)

9 fig = m.plot\_components(forecast)



Both of the seasonalities now show up in the components plots above. We can see that during the on-season when games are played every Sunday, there are large increases on Sunday and Monday that are completely absent during the off-season.

# Prior scale for holidays and seasonality假日和季节性的先验尺度

If you find that the holidays are overfitting, you can adjust their prior scale to smooth them using the parameter

***holidays\_prior\_scale***.

By default this parameter is 10, which provides very little regularization. Reducing this

使用***holidays\_prior\_scale***参数来规定假日的先验尺度，以调整训练的拟合度，使得其更加平滑。

默认情况下，此参数为10，正则化很少。

减少这个参数可以减弱节日效应:

1. # Python
2. m = Prophet(holidays=holidays, holidays\_prior\_scale=0.05).fit(df) 3 forecast = m.predict(future)
3. forecast[(forecast['playoff'] + forecast['superbowl']).abs() > 0][
4. ['ds', 'playoff', 'superbowl']][-10:]

**2190**2014-02-02 1.206086 0.964914

**2191**2014-02-03 1.852077 0.992634

**2532**2015-01-11 1.206086 0.000000

**2533**2015-01-12 1.852077 0.000000

**2901**2016-01-17 1.206086 0.000000

**2902**2016-01-18 1.852077 0.000000

**2908**2016-01-24 1.206086 0.000000

**2909**2016-01-25 1.852077 0.000000

**2922**2016-02-07 1.206086 0.964914

**2923**2016-02-08 1.852077 0.992634

**superbowl**

**playoff**

**ds**

The magnitude of the holiday effect has been reduced compared to before, especially for superbowls, which had the fewest observations. There is a parameter ***seasonality\_prior\_scale*** which similarly adjusts the extent to which the seasonality model will fit the data.

与以前相比，节日效应的程度有所降低，特别是对于观测次数最少的超级碗来说。还有一个参数***seasonality\_prior\_scale***，它类似地调整了季节性模型拟合数据的程度。

Prior scales can be set separately for individual holidays by including a column ***prior\_scale*** in the holidays dataframe. Prior scales for individual seasonalities can be passed as an argument to ***add\_seasonality***. 通过在节假日数据框架中包含一个列***prior\_scale***，可以为单个节假日单独设置优先比例。单独季节性的先验量表可以作为参数传递给***add\_seasonality。***

For instance, the prior scale for just weekly seasonality can be set using: 例如，可以使用以下命令设置仅为每周季节性的先验刻度

1. # Python
2. m = Prophet()
3. m.add\_seasonality(
4. name='weekly', period=7, fourier\_order=3, prior\_scale=0.1)

# Additional regressors

Additional regressors can be added to the linear part of the model using the ***add\_regressor()***  method or function. 可以使用***add\_regressor***方法或函数向模型的线性部分添加额外的回归量

A column with the regressor value will need to be present in both the fitting and prediction dataframes. 带有回归值的列需要同时出现在拟合和预测数据框架中。

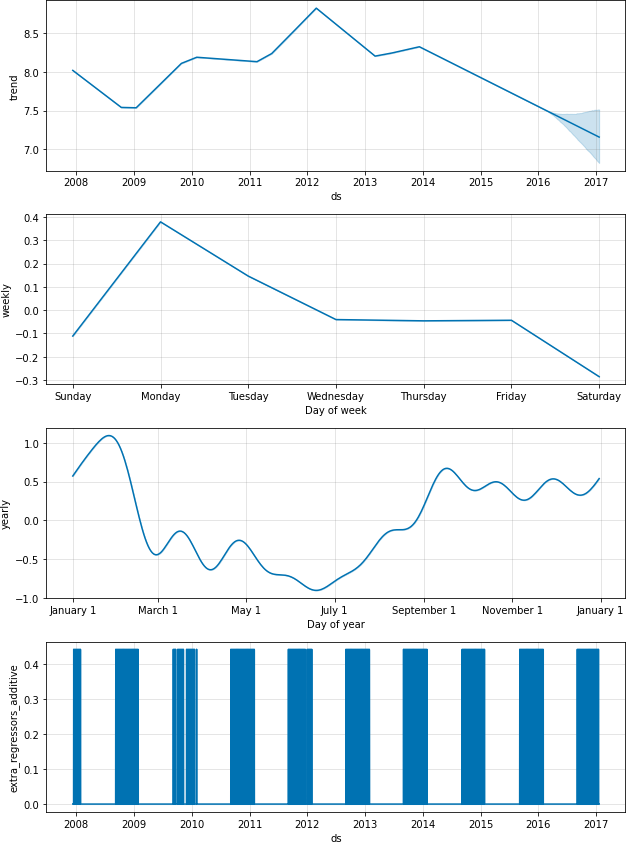
For example, we can add an additional effect on Sundays during the NFL season. On the components plot, this effect will show up in the ‘extra\_regressors’ plot: 例如，我们可以在NFL赛季的周日添加一个额外的效果。在分量图中，这种效应将在“extra\_regressors”图中显示:

1. # Python
2. def nfl\_sunday(ds):
3. date = pd.to\_datetime(ds)
4. if date.weekday() == 6 and (date.month > 8 or date.month < 2):
5. return 1
6. else:
7. return 0
8. df['nfl\_sunday'] = df['ds'].apply(nfl\_sunday) 9
9. m = Prophet()
10. m.add\_regressor('nfl\_sunday')
11. m.fit(df)

13

14 future['nfl\_sunday'] = future['ds'].apply(nfl\_sunday) 15

1. forecast = m.predict(future)
2. fig = m.plot\_components(forecast)



NFL Sundays could also have been handled using the “***holidays***” interface described above, by creating a list of past and future NFL Sundays. **NFL sundays**也可以使用上面描述的“假日”界面处理，通过创建过去和未来NFL周日的列表。

The add\_regressor function provides a more general interface for defining extra linear regressors, and in particular does not require that the regressor be a binary indicator. Another time series could be used as a regressor, although its future values would have to be known. add\_regressor函数为定义额外的线性回归函数提供了更通用的接口，特别是不要求回归函数是二进制指示符。另一个时间序列可以用作回归函数，尽管它的未来值必须是已知的。

[**This notebook**](../Jupyter%20Notebook%20Viewer.pdf) **shows an example of using weather factors as extra regressors in a forecast of bicycle usage, and provides an excellent illustration of how other time series can be included as extra regressors. 这本笔记本展示了一个在自行车使用预测中使用天气因素作为额外回归量的例子，并提供了一个很好的说明如何将其他时间序列作为额外回归量。**

The add\_regressor function has optional arguments for specifying the prior scale (holiday prior scale is used by default) add\_regressor函数有可选参数，用于指定优先比例(默认使用假日优先比例)

and whether or not the regressor is standardized - see the docstring with help(Prophet.add\_regressor) in Python and

?add\_regressor in R. Note that regressors must be added prior to model fitting. Prophet will also raise an error if the regressor is constant throughout the history, since there is nothing to fit from it. 注意，必须在模型拟合之前添加回归量。如果回归量在整个历史中是恒定的，先知也会提出一个错误，因为没有什么可以从它中拟合。

The extra regressor must be known for both the history and for future dates. It thus must either be something that has known future values (such as nfl\_sunday), or something that has separately been forecasted elsewhere. The weather regressors used in the notebook linked above is a good example of an extra regressor that has forecasts that can be used for future values. One can also use as a regressor another time series that has been forecasted with a time series model, such as Prophet. For instance, if r(t) is included as a regressor for y(t), Prophet can be used to forecast r(t) and then that forecast can be plugged in as the future values when forecasting y(t). A note of caution around this approach: This will probably not be useful unless r(t) is somehow easier to forecast then y(t). This is because error in the forecast of r(t) will produce error in the forecast of y(t). One setting where this can be useful is in hierarchical time series, where there is top-level forecast that has higher signal-to-noise and is thus easier to forecast. Its forecast can be included in the forecast for each lower-level series. 额外的回归量必须同时知道历史和未来的日期。因此，它必须要么是已知的未来值(例如nfl\_sunday)，要么是在其他地方单独预测的值。上面链接的笔记本中使用的天气回归器是一个额外回归器的很好的例子，它可以用于预测未来的值。也可以使用另一个时间序列作为回归量，该时间序列模型已经预测过，例如Prophet。例如，如果r(t)被包括为y(t)的回归量，则可以使用Prophet来预测r(t)，然后在预测y(t)时，该预测可以作为未来的值插入。关于这种方法需要注意的是:除非r(t)比y(t)更容易预测，否则这种方法可能没有用处。这是因为r(t)的预测误差会导致y(t)的预测误差。这种方法很有用的一种情况是在分层时间序列中，其中有顶级预测，具有更高的信噪比，因此更容易预测。它的预测可以包含在每个较低级别系列的预测中。

Extra regressors are put in the linear component of the model, so the underlying model is that the time series depends on the extra regressor as either an additive or multiplicative factor (see the next section for multiplicativity).

**Coefficients of additional regressors附加回归系数**

To extract the beta coefficients of the extra regressors, use the utility function regressor\_coefficients (from prophet.utilities import regressor\_coefficients in Python, prophet::regressor\_coefficients in R) on the fitted model. The estimated beta coefficient for each regressor roughly represents the increase in prediction value for a unit increase in the regressor value (note that the coefficients returned are always on the scale of the original data). If mcmc\_samples is specified, a credible interval for each coefficient is also returned, which can help identify whether each regressor is “statistically significant”.

[Edit on GitHub](https://github.com/facebook/prophet/blob/main/docs/_docs/seasonality%2C_holiday_effects%2C_and_regressors.md)

**Facebook Open Source**

[Open Source Projects](https://code.facebook.com/projects/) [GitHub](https://github.com/facebook/) [Twitter](https://twitter.com/fbOpenSource) [Privacy](https://opensource.facebook.com/legal/privacy/) [Terms](https://opensource.facebook.com/legal/terms/) [Contribute to this project on GitHub](https://github.com/facebook/prophet)